


```

LL                      IIIIIIII                      SSSSSSSSSS
LL                      IIIIIIII                        SSSSSSSSSS
LL                      IIII                             SS
LL                      IIII                             SS
LL                      IIII                             SS
LL                      IIII                             SS
LL                      IIII                             SSSSSSSS
LL                      IIII                             SSSSSSSS
LL                      IIII                             SS
LL                      IIII                             SS
LL                      IIII                             SS
LL                      IIII                             SS
LLLLLLLLLLLLLLLL      IIIIIIIII                      SSSSSSSSSS
LLLLLLLLLLLLLLLL      IIIIIIIII                      SSSSSSSSSS

```

(2)	44	Edit History
(3)	77	DECLARATIONS
(4)	159	OTSS\$CVT_D_T - Convert D floating to text
(5)	247	OTSS\$CVT_D_T_R8
(6)	324	Numeric conversion routines
(7)	415	Character formatting routines

```
0000 1 .TITLE OTSS$CVTDT
0000 2 .IDENT /1-017/
0000 3
0000 4
0000 5 *****
0000 6 *
0000 7 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 8 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 9 * ALL RIGHTS RESERVED.
0000 10 *
0000 11 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 12 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 13 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 14 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 15 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 16 * TRANSFERRED.
0000 17 *
0000 18 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 19 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 20 * CORPORATION.
0000 21 *
0000 22 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 23 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 24 *
0000 25 *
0000 26 *****
0000 27
0000 28
0000 29 ++
0000 30 FACILITY: Language-independent Support Library
0000 31
0000 32 ABSTRACT:
0000 33
0000 34 A routine to convert an F or D-floating value to a string of
0000 35 ASCII digits and an exponent. It is meant to be used as
0000 36 a base for floating point output conversion routines.
0000 37
0000 38 ENVIRONMENT: User Mode, AST Reentrant
0000 39
0000 40 --
0000 41 AUTHOR: Steven B. Lionel, CREATION DATE: 24-May-1979
0000 42
```


Edit History

```
0000 44 .SBTTL Edit History
0000 45 :
0000 46 : 1-001 - Original. Numeric conversion algorithm by Trygve Fossum.
0000 47 : SBL 24-May-1979
0000 48 : 1-002 - Make routine an R8 so as to conform with OTSS$CVTDT. SBL 3-Jul-1979
0000 49 : 1-003 - Add extra longword to stack frame to prevent clobbering of
0000 50 : saved info. SBL 8-Jul-1979
0000 51 : 1-004 - Don't use R9 or R10 at all. SBL 11-Jul-1979
0000 52 : 1-005 - On right-rounding to zero, don't change the sign. SBL 16-Jul-1979
0000 53 : 1-006 - Fix typo in stack frame setup. SBL 23-July-1979
0000 54 : 1-007 - Modify rounding algorithm so that if RT_RND would cause
0000 55 : rounding to the right of the number of significant digits,
0000 56 : the latter is used instead. This is at the request of
0000 57 : BASIC - the situation can not occur in FORTRAN. SBL 27-Jul-1979
0000 58 : 1-008 - Clear 96 bits ahead of fraction instead of 63. SBL 30-July-1979
0000 59 : 1-009 - Speed improvements. Clear 64 bits ahead of fraction. Use
0000 60 : register in inner convert loop. SBL 21-Jan-1980
0000 61 : 1-010 - Compute number of fraction longwords correctly at INIT_FRACT,
0000 62 : to assure accurate low-order digits. JAW 21-Jul-1981
0000 63 : 1-011 - Make sure all bits between significand and binary point are
0000 64 : cleared when value is an integer, to assure accurate low-order
0000 65 : digits. JAW 26-Jul-1981
0000 66 : 1-012 - If we find a reserved operand, return zero if it doesn't get
0000 67 : replaced by a non-reserved value. SBL 29-Oct-81
0000 68 : 1-013 - Add entry for Ffloating. SBL 29-Oct-1982
0000 69 : 1-014 - Remove CVTFD instruction from OTSS$CVT_F T R8. SBL 27-Apr-1983
0000 70 : 1-015 - Fix bug introduced by 1-014. SBL 17-May-1983
0000 71 : 1-016 - Removed the CVTLP, CVTPS, and SKPC instructions to improve the
0000 72 : performance of this routine. Instead, EDIV instructions were
0000 73 : used. I also fixed a couple of comments. JCW 31-OCT-1983
0000 74 : 1-017 - Move tables after PSECT definition. LEB 22-Mar-1984
0000 75 :
```

DECLARATIONS

```

0000 77      .SBTTL DECLARATIONS
0000 78      :
0000 79      INCLUDE FILES:
0000 80      :
0000 81      :
0000 82      :
0000 83      EXTERNAL DECLARATIONS:
0000 84      :
0000 85      .DSABL GBL                      : Prevent undeclared
0000 86      :                               : symbols from being
0000 87      :                               : automatically global.
0000 88      :
0000 89      :
0000 90      MACROS:
0000 91      :
0000 92      :
0000 93      :
0000 94      EQUATED SYMBOLS:
0000 95      :
0000 96      :
0000 97      :
0000 98      PSECT DECLARATIONS:
0000 99      :
00000000 100      .PSECT _OTSS$CODE PIC, USR, CON, REL, LCL, SHR, -
0000 101      EXE, RD, NOWRT, LONG
0000 102      :
0000 103      :
0000 104      OWN STORAGE:
0000 105      :
0000 106      CONSTANTS
0000 107      :
0000 108      :
0000 109      ASCII_ZEROES:
30303030 30303030 0000 110      .QUAD      ^X3030303030303030      : 8 copies of the character 0
0008 111      :
3430 3330 3230 3130 3030 0008 112      TABLE: .WORD      ^X3030, ^X3130, ^X3230, ^X3330, ^X3430
3930 3830 3730 3630 3530 0012 113      .WORD      ^X3530, ^X3630, ^X3730, ^X3830, ^X3930
3431 3331 3231 3131 3031 001C 114      .WORD      ^X3031, ^X3131, ^X3231, ^X3331, ^X3431
3931 3831 3731 3631 3531 0026 115      .WORD      ^X3531, ^X3631, ^X3731, ^X3831, ^X3931
3432 3332 3232 3132 3032 0030 116      .WORD      ^X3032, ^X3132, ^X3232, ^X3332, ^X3432
3932 3832 3732 3632 3532 003A 117      .WORD      ^X3532, ^X3632, ^X3732, ^X3832, ^X3932
3433 3333 3233 3133 3033 0044 118      .WORD      ^X3033, ^X3133, ^X3233, ^X3333, ^X3433
3933 3833 3733 3633 3533 004E 119      .WORD      ^X3533, ^X3633, ^X3733, ^X3833, ^X3933
3434 3334 3234 3134 3034 0058 120      .WORD      ^X3034, ^X3134, ^X3234, ^X3334, ^X3434
3934 3834 3734 3634 3534 0062 121      .WORD      ^X3534, ^X3634, ^X3734, ^X3834, ^X3934
3435 3335 3235 3135 3035 006C 122      .WORD      ^X3035, ^X3135, ^X3235, ^X3335, ^X3435
3935 3835 3735 3635 3535 0076 123      .WORD      ^X3535, ^X3635, ^X3735, ^X3835, ^X3935
3436 3336 3236 3136 3036 0080 124      .WORD      ^X3036, ^X3136, ^X3236, ^X3336, ^X3436
3936 3836 3736 3636 3536 008A 125      .WORD      ^X3536, ^X3636, ^X3736, ^X3836, ^X3936
3437 3337 3237 3137 3037 0094 126      .WORD      ^X3037, ^X3137, ^X3237, ^X3337, ^X3437
3937 3837 3737 3637 3537 009E 127      .WORD      ^X3537, ^X3637, ^X3737, ^X3837, ^X3937
3438 3338 3238 3138 3038 00A8 128      .WORD      ^X3038, ^X3138, ^X3238, ^X3338, ^X3438
3938 3838 3738 3638 3538 00B2 129      .WORD      ^X3538, ^X3638, ^X3738, ^X3838, ^X3938
3439 3339 3239 3139 3039 00BC 130      .WORD      ^X3039, ^X3139, ^X3239, ^X3339, ^X3439
3939 3839 3739 3639 3539 00C6 131      .WORD      ^X3539, ^X3639, ^X3739, ^X3839, ^X3939
00D0 132      :
00D0 133      : Stack frame offsets from R7

```

DECLARATIONS

```

00D0 134 :: Common frame for kernel convert routines
FFFFFFFF8 00D0 135 PACKED = -8 ; Temp for packed representation
FFFFFFFF4 00D0 136 FLAGS = PACKED - 4 ; Flags for outer and inner routines
FFFFFFFF0 00D0 137 SIG DIGITS = FLAGS - 4 ; Significant digits
FFFFFFEC 00D0 138 STRING_ADDR = SIG DIGITS - 4 ; Address of temp string
FFFFFFE8 00D0 139 SIGN = -STRING_ADDR - 4 ; Sign
FFFFFFE4 00D0 140 DEC_EXP = SIGN - 4 ; Decimal exponent
FFFFFFE0 00D0 141 OFFSET = DEC_EXP - 4 ; Offset
FFFFFFDC 00D0 142 RT_RND = OFFSET - 4 ; Right round point
FFFFFFDC 00D0 143 COMMON_FRAME = RT_RND ; Common frame size
00D0 144
00D0 145 ::+
00D0 146 :: Inner routine frame pointed to by R8 during conversion
00D0 147 ::-
FFFFFFFF0 00D0 148 INT_HI = -16 ; Highest integer part
FFFFFFE4 00D0 149 BIN_PT = INT_HI - 12 ; Binary point
FFFFFFC8 00D0 150 FRACT_LIM = BIN_PT - 28 ; Lowest fraction bits
FFFFFFB4 00D0 151 DIGITS = FRACT_LIM - 20 ; Digits radix 10**9
FFFFFFB0 00D0 152 BIN_EXP = DIGITS - 4 ; Saved binary exponent
FFFFFFAC 00D0 153 LONG_COUNT = BIN_EXP - 4 ; Longword count
FFFFFFA8 00D0 154 TEMP = LONG_COUNT - 4 ; Temporary
FFFFFFA8 00D0 155 LOCAL_FRAME = TEMP ; Local frame size
00D0 156
00D0 157

```


OTSS\$CVT_D_T - Convert D floating to te

```

00D0 159 .SBTTL OTSS$CVT_D_T - Convert D floating to text
00D0 160
00D0 161 **
00D0 162 FUNCTIONAL DESCRIPTION:
00D0 163
00D0 164 This routine converts a D-floating point value to a string
00D0 165 of ASCII digits. It is intended to form the base of a
00D0 166 language's floating point output conversion routine.
00D0 167
00D0 168 OTSS$CVT_F_T_R8 converts F-floating.
00D0 169
00D0 170 CALLING SEQUENCE:
00D0 171
00D0 172 MOVAB common_frame, R1 ; See common_frame definition above
00D0 173 MOVL string_length, STRING_LEN(R1)
00D0 174 MOVL string_address, STRING_ADDR(R1)
00D0 175 MOVL sig_digits, SIG_DIGITS(R1)
00D0 176 MOVL user_flags, FLAGS(R1)
00D0 177 MOVL rt_round, RT_RND(R1) ; Optional
00D0 178 MOVAB value, R0
00D0 179 JSB OTSS$CVT_D_T_R8 or OTSS$CVT_F_T_R8
00D0 180 ; outputs are:
00D0 181 ; R1 = unchanged
00D0 182 ; OFFSET(R1) - offset
00D0 183 ; DEC EXP(R1) - decimal exponent
00D0 184 ; SIGN(R1) - sign
00D0 185
00D0 186 INPUT PARAMETERS:
00D0 187
00D0 188 VALUE ; F or D-floating value to be converted
00D0 189 SIG_DIGITS(R1) ; Number of significant digits to
00D0 190 ; generate. If neither V_TRUNCATE
00D0 191 ; or V_ROUND_RIGHT is set, the
00D0 192 ; value will be rounded to this
00D0 193 ; many digits.
00D0 194 ; Caller supplied flags:
00D0 195 ; V_TRUNCATE = 24
00D0 196 ; V_ROUND_RIGHT = 25
00D0 197 ;
00D0 198 RT_RND(R1) ; Number of places to the right
00D0 199 ; of the decimal point to round
00D0 200 ; after. Ignored if V_ROUND_RIGHT
00D0 201 ; is clear. The rounding takes
00D0 202 ; place after the specified number
00D0 203 ; of significant digits if that
00D0 204 ; would be farther to the left.
00D0 205
00D0 206 IMPLICIT INPUTS:
00D0 207
00D0 208 NONE
00D0 209
00D0 210 OUTPUT PARAMETERS:
00D0 211
00D0 212 out_string ; String with result. It will
00D0 213 ; Not have valid digits after the
00D0 214 ; requested number of significant
00D0 215 ; digits.
00D0 216 ; The length MUST be at least:

```

00000018
00000019


```

00D0 216 ;      offset      ; (9*INT((sig_digits+17)/9))+2
00D0 217 ;      ;          ; The offset into out_string at
00D0 218 ;      ;          ; which the first significant digit
00D0 219 ;      ;          ; may be found. It is guaranteed
00D0 220 ;      ;          ; to be between 0 and 9.
00D0 221 ;      exponent    ; The signed decimal exponent of
00D0 222 ;      ;          ; the value, assuming a radix point
00D0 223 ;      ;          ; immediately to the left of the
00D0 224 ;      ;          ; most significant digit.
00D0 225 ;      sign        ; -1 if the value is negative
00D0 226 ;      ;          ; 0 if the value is zero
00D0 227 ;      ;          ; 1 if the value is positive
00D0 228 :
00D0 229 : IMPLICIT OUTPUTS:
00D0 230 :
00D0 231 :     NONE
00D0 232 :
00D0 233 : SIDE EFFECTS:
00D0 234 :
00D0 235 :     Alters registers R0 through R8.
00D0 236 :
00D0 237 :     SS$_ROPRAND - If the value is a reserved operand
00D0 238 :     SS$_ACCVIO  , or other nasty errors if the length of
00D0 239 :                   out_string is not enough (see formula above).
00D0 240 :                   This routine does not check the length, it
00D0 241 :                   is up to the caller to insure the correct
00D0 242 :                   length is present.
00D0 243 :
00D0 244 : --
00D0 245 :

```

OTSS\$CVT_D_T_R8

```
00D0 247 .SBTTL OTSS$CVT_D_T_R8
00D0 248
00D0 249
00D0 250 :+ JSB entry point
00D0 251 :-
00D0 252
00D0 253 OTSS$CVT_F_T_R8::
57 51 D0 00D0 254 MOVL R1, R7 : Use R7 as common frame pointer
50 51 D4 00D3 255 CLRL R1 : Clear high part of value
50 60 50 00D5 256 MOVF (R0), R0 : Fetch and test for zero
06 11 00D8 257 BRB COMMON_FD : Join common code
00DA 258
00DA 259 OTSS$CVT_D_T_R8::
57 51 D0 00DA 260 MOVL R1, R7 : Use R7 as common frame pointer
50 60 70 00DD 261 MOVD (R0), R0 : Fetch and test for zero
00E0 262 : and for reserved operand
00E0 263 COMMON_FD:
0E 14 00E0 264 BGTR VAL_POS : Value is positive
06 19 00E2 265 BLSS VAL_NEG : Value is negative
7D 10 00E4 266 BSBB ZERO : Value is zero
51 57 D0 00E6 267 MOVL R7, R1 : Restore R1
05 00E9 268 RSB : Return to caller
00EA 269
00EA 270 VAL_NEG:
EB A7 01 CE 00EA 271 MNEGL #1, SIGN(R7) : Set negative sign
04 11 00EE 272 BRB EXTRACT : Continue
00F0 273 VAL_POS:
EB A7 01 D0 00F0 274 MOVL #1, SIGN(R7) : Set positive sign
00F4 275
00F4 276 EXTRACT:
58 5E D0 00F4 277 MOVL SP, R8 : R8 points to local frame
5E AB AE 9E 00F7 278 MOVAB LOCAL_FRAME(SP), SP : Set up local frame
54 E4 AB 9E 00FB 279 MOVAB BIN_PT(R8), R4 : R4 points to binary point
52 50 08 07 EF 00FF 280 EXTZV #7, #8, R0, R2 : Extract exponent
50 50 10 9C 0104 281 BEQL ZERO : Still reserved operand; give up
51 51 10 9C 0106 282 ROTL #16, R0, R0 : Make into proper fraction
52 00000080 8F C2 010A 283 ROTL #16, R1, R1
05 18 010E 284 SUBL2 #128, R2
FB A4 7C 0115 285 BGEQ 10$ : Remove bias
02 11 0117 286 CLRQ -8(R4) : If value is less than 1,
64 7C 011A 287 : clear some fraction bits
011E 288 : in case value is < 2**64.
011E 289 10$: BRB 20$
011E 290 20$: INSV #0, R2, #32, -11(R4)
F5 A4 20 52 00 F0 011E 291 INSV R1, R2, #32, -7(R4)
F9 A4 20 52 51 F0 0124 292 BISL2 #X800000, R0
50 00800000 8F C8 012A 293 INSV R0, R2, #24, -3(R4)
FD A4 18 52 50 F0 0131 294 INSV #0, R2, #32, (R4)
64 20 52 00 F0 0137 295 INSV #0, R2, #32, +4(R4)
04 A4 20 52 00 F0 013C 296 MOVL R2, BIN_EXP(R8) : Save binary exponent
B0 AB 52 D0 0142 297 BLEQ FRACT_ONLY : If less than 1...
2D 15 0146 298 MOVAB DIGITS(R8), R6 : R6 points to scratch area
56 B4 AB 9E 0148 300 PUSH R7 : Save R7 so we can use as temp
55 52 FB 8F DD 014C 301 ASHL #5, R2, R5 : How many integer longwords?
AC AB 55 D0 0153 302 MOVL R5, LONG_COUNT(R8)
303
```

				OTSS\$CVT_D_T_R8			
		03	07	15	0157	304	BLEQ ONE_LONG ; 1 longword
			55	91	0159	305	CMPB R5, #3
			29	19	015C	306	BLSS INT_LOOP ; 2 or 3 longwords
			1C	11	015E	307	BRB FOUR_LONG ; Four longwords
					0160	308	
		0078		31	0160	309	ONE_LONG:
					0160	310	BRW INT_NEXT
					0163	311	
					0163	312	ZERO:
61	FD A7	30	51	EC A7	D0	0163	313
			6E	00	2C	0167	314
				EO A7	7C	016E	315
				EB A7	D4	0171	316
					05	0174	317
						0175	318
						0175	319
		AC AB	01	CE	0175	320	FRACT_ONLY:
			00E0	31	0179	321	MNEGL
					017C	322	BRW
							#1, LONG_COUNT(R8) ; To note that there is no integer part
							FORMAT ; Go directly to formatter

Numeric conversion routines

.SBTTL Numeric conversion routines

```

017C 324
017C 325
017C 326
017C 327
017C 328
017C 329
017C 330
017C 331
017C 332
017C 333
017E 334
0182 335
0185 336
0187 337
0187 338
018C 339
018F 340
0191 341
0193 342
0196 343
0198 344
019B 345
019B 346
01A4 347
01A7 348
01A7 349
01AA 350
01B2 351
01B4 352
01B7 353
01C0 354
01C0 355
01C7 356
01C9 357
01CC 358
01D5 359
01D5 360
01D8 361
01DB 362
01DE 363
01DE 364
01E2 365
01E6 366
01F0 367
01F2 368
01F4 369
01F8 370
01FA 371
01FC 372
01FE 373
0202 374
0205 375
0207 376
0207 377
0207 378
0207 379
0207 380

54 51 D4 017C 324
FO A8 DE 017C 325
50 64 D0 017E 334
14 11 0182 335
54 E4 A845 DE 0187 337
04 A4 D5 018C 338
07 13 018F 339
55 D6 0191 340
AC A8 D6 0193 341
EF 11 0196 342
50 64 7D 0198 343
51 64 50 3B9ACA00 8F 7B 019B 344
04 A4 D4 019B 345
57 55 D0 01A4 346
53 51 1DCD6500 8F C3 01A7 347
15 19 01A7 348
52 74 D0 01AA 349
51 64 52 3B9ACA00 8F 7B 01B2 351
01C0 352
64 80000000 8F C8 01B4 353
0C 11 01B7 354
50 74 D0 01C0 355
51 64 50 3B9ACA00 8F 7B 01C7 356
01C9 357
D2 57 F5 01CC 358
86 51 D0 01D5 359
A9 55 F5 01D5 360
01D8 361
01DB 362
52 AC A8 D0 01DE 363
50 E4 A8 7D 01DE 364
86 04 A6 50 3B9ACA00 8F 7B 01E2 365
02 13 01E6 366
52 D6 01F0 367
B4 A842 D5 01F2 368
04 12 01F4 369
52 D7 01F8 370
F6 11 01FA 371
AC A8 52 D0 01FC 372
57 8E D0 01FE 373
55 11 0202 374
0205 375
0207 376
0207 377
0207 378
0207 379
0207 380

: This is the portion which converts the integer part of the value
: to 1-5 longwords of radix 10**9. This is done by repeated division
: by 10**9.
FOUR_LONG:
CLRL R1 : High part of dividend
MOVAL INT_HI(R8), R4 : Use R4 as address pointer
MOVL (R4), R0 : Low part of dividend
BRB INT_DIV
INT_LOOP:
MOVAL BIN_PT(R8)[R5], R4 : Get address pointer
TSTL 4(R4) : Are we missing some bits?
BEQL 10$ : No if zero
INCL R5 : Back up one longword
INCL LONG_COUNT(R8) : Bump longword counter
BRB INT_LOOP : And try again
MOVQ (R4), R0 : Get first quadword of dividend
10$:
INT_DIV:
EDIV #^D1000000000,R0,(R4),R1
CLRL 4(R4) : Since this is really a
: quadword quotient, zero the
: higher longword.
: R7 is inner loop counter
MOVL R5, R7
30$:
SUBL3 #^D500000000,R1,R3
BLSS 40$ : Is this dividend too large ?
MOVL -(R4), R2 : Skip adjustment if not
EDIV #^D1000000000,R2,(R4),R1 : Low part of dividend
: Divide by 10**9
BISL #^X80000000,(R4) : Set high bit
BRB 60$
40$:
MOVL -(R4), R0 : Get low part of dividend
EDIV #^D1000000000,R0,(R4),R1 : Divide and store result in (R4)
60$:
SOBGTR R7, 30$ : Loop back
MOVL R1, (R6)+ : Store result on stack
SOBGTR R5, INT_LOOP : Loop back if not done
INT_NEXT:
MOVL LONG_COUNT(R8), R2
MOVQ BIN_PT(R8), R0 : Low part of next dividend
EDIV #^D1000000000,R0,4(R6),(R6)+
BEQL 10$ : Branch if high longword is 0
INCL R2 : Convert one more longword
10$:
TSTL DIGITS(R8)[R2] : Find first non-zero longword
BNEQ 20$ : Found. Go format them.
DECL R2 : Not found. Try next one.
BRB 10$
20$:
MOVL R2, LONG_COUNT(R8) : Save longword count
MOVL (SP)+, R7 : Restore R7 from where saved
BRB FORMAT
: This routine initializes the pointer for getting fraction digits.
: The number of fraction longwords is calculated and is stored in
: LONG_COUNT(R8) for future calls.

```

Numeric conversion routines

PC	Op	Op2	Op3	Op4	Op5	Op6	Op7	Op8	Op9	Op10	Op11	Op12	Op13	Op14	Op15	Op16	Op17	Op18	Op19	Op20	Op21	Op22	Op23	Op24	Op25	Op26	Op27	Op28	Op29	Op30	Op31	Op32	Op33	Op34	Op35	Op36	Op37	Op38	Op39	Op40	Op41	Op42	Op43	Op44	Op45	Op46	Op47	Op48	Op49	Op50	Op51	Op52	Op53	Op54	Op55	Op56	Op57	Op58	Op59	Op60	Op61	Op62	Op63	Op64	Op65	Op66	Op67	Op68	Op69	Op70	Op71	Op72	Op73	Op74	Op75	Op76	Op77	Op78	Op79	Op80	Op81	Op82	Op83	Op84	Op85	Op86	Op87	Op88	Op89	Op90	Op91	Op92	Op93	Op94	Op95	Op96	Op97	Op98	Op99	Op100	Op101	Op102	Op103	Op104	Op105	Op106	Op107	Op108	Op109	Op110	Op111	Op112	Op113	Op114	Op115	Op116	Op117	Op118	Op119	Op120	Op121	Op122	Op123	Op124	Op125	Op126	Op127	Op128	Op129	Op130	Op131	Op132	Op133	Op134	Op135	Op136	Op137	Op138	Op139	Op140	Op141	Op142	Op143	Op144	Op145	Op146	Op147	Op148	Op149	Op150	Op151	Op152	Op153	Op154	Op155	Op156	Op157	Op158	Op159	Op160	Op161	Op162	Op163	Op164	Op165	Op166	Op167	Op168	Op169	Op170	Op171	Op172	Op173	Op174	Op175	Op176	Op177	Op178	Op179	Op180	Op181	Op182	Op183	Op184	Op185	Op186	Op187	Op188	Op189	Op190	Op191	Op192	Op193	Op194	Op195	Op196	Op197	Op198	Op199	Op200	Op201	Op202	Op203	Op204	Op205	Op206	Op207	Op208	Op209	Op210	Op211	Op212	Op213	Op214	Op215	Op216	Op217	Op218	Op219	Op220	Op221	Op222	Op223	Op224	Op225	Op226	Op227	Op228	Op229	Op230	Op231	Op232	Op233	Op234	Op235	Op236	Op237	Op238	Op239	Op240	Op241	Op242	Op243	Op244	Op245	Op246	Op247	Op248	Op249	Op250	Op251	Op252	Op253	Op254	Op255	Op256	Op257	Op258	Op259	Op260	Op261	Op262	Op263	Op264	Op265	Op266	Op267	Op268	Op269	Op270	Op271	Op272	Op273	Op274	Op275	Op276	Op277	Op278	Op279	Op280	Op281	Op282	Op283	Op284	Op285	Op286	Op287	Op288	Op289	Op290	Op291	Op292	Op293	Op294	Op295	Op296	Op297	Op298	Op299	Op300	Op301	Op302	Op303	Op304	Op305	Op306	Op307	Op308	Op309	Op310	Op311	Op312	Op313	Op314	Op315	Op316	Op317	Op318	Op319	Op320	Op321	Op322	Op323	Op324	Op325	Op326	Op327	Op328	Op329	Op330	Op331	Op332	Op333	Op334	Op335	Op336	Op337	Op338	Op339	Op340	Op341	Op342	Op343	Op344	Op345	Op346	Op347	Op348	Op349	Op350	Op351	Op352	Op353	Op354	Op355	Op356	Op357	Op358	Op359	Op360	Op361	Op362	Op363	Op364	Op365	Op366	Op367	Op368	Op369	Op370	Op371	Op372	Op373	Op374	Op375	Op376	Op377	Op378	Op379	Op380	Op381	Op382	Op383	Op384	Op385	Op386	Op387	Op388	Op389	Op390	Op391	Op392	Op393	Op394	Op395	Op396	Op397	Op398	Op399	Op400	Op401	Op402	Op403	Op404	Op405	Op406	Op407	Op408	Op409	Op410	Op411	Op412	Op413	Op414	Op415	Op416	Op417	Op418	Op419
----	----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------

Character formatting routines

```
025C 415 .SBTTL Character formatting routines
025C 416
025C 417 :+ After all the integer portion of the value has been converted to
025C 418 : longwords and stored, the integer part is then converted to
025C 419 : characters and the fraction part, if any, is converted.
025C 420 :-
025C 421
025C 422 FORMAT:
025C 423
025C 424 MOVL STRING_ADDR(R7), R5 : Get string address
0260 425 MOVW #^A/O/, (R5)+ : Set first character to '0'
0263 426 ADDL3 #1, SIG_DIGITS(R7), R6 : Generate at least one extra digit
0268 427 MOVL LONG_COUNT(R8), R0 : How many integer longwords?
026C 428 BGEQ 1$ : If none, skip this part
026E 429 BRW NO_INT : R5 will store least signif digit
0271 430 ADDL2 #9, R5 : (lsd) in the high order byte.
0274 431
0274 432 MOVL R5, R3 : save the old address
0277 433 MOVW ASCII_ZEROES, -9(R5) : Initialize the string to contain 30's
027D 434 : the 9th byte will be filled below
027D 435
027D 436 MOVL DIGITS(R8)[R0], R1 : R1/R2 must be a quadword for
0282 437 CLRL R2 : the EDIV
0284 438 EDIV #100, R1, R1, R4 : extract two lsd
028D 439 BEQL 60$ :
028F 440 MOVW TABLE[R4], -(R5) : load correct char rep of the 2 digits
0295 441 EDIV #100, R1, R1, R4 : extract two lsd
029E 442 BEQL 60$ :
02A0 443 MOVW TABLE[R4], -(R5) : load correct char rep of the 2 digits
02A6 444 EDIV #100, R1, R1, R4 : extract two lsd
02AF 445 BEQL 60$ :
02B1 446 MOVW TABLE[R4], -(R5) : load correct char rep of the 2 digits
02B7 447 EDIV #100, R1, R1, R4 : extract two lsd
02C0 448 MOVW TABLE[R4], -(R5) : load correct char rep of the 2 digits
02C6 449 ADDB3 #^A/O/, R1, -(R5) : character rep needed for last number
02CA 450
02CA 451 : Numbers are stored as characters as follows: low order byte is the most
02CA 452 : significant digit (character), while the high order byte is the least signif
02CA 453 : digit (character). The storage took place from the high order digit to the
02CA 454 : low order digit. Since we used an EDIV by 100, 0, 1, or 2 zeroes may be
02CA 455 : located at (R5). R0 is to contain the number of nonzero digits (not char 30)
02CA 456 : between (R3) and (R5). If R1<>0 then R0=9. If R1=0, there is at least one
02CA 457 : zero at (R5) and possibly another at (R5)-1. For example, 12 --> 323130
02CA 458 : while 102 --> 3230313030.
02CA 459
02CA 460
02CE 461
02D0 462
02D2 463
02D4 464
02D8 465
02DA 466
02DC 467
02DC 468
02E1 469
02E6 470
02EB 471
02EE 472
02F1 473
02F1 474
02F1 475
02F1 476
02F1 477
02F1 478
02F1 479
02F1 480
02F1 481
02F1 482
02F1 483
02F1 484
02F1 485
02F1 486
02F1 487
02F1 488
02F1 489
02F1 490
02F1 491
02F1 492
02F1 493
02F1 494
02F1 495
02F1 496
02F1 497
02F1 498
02F1 499
02F1 500
02F1 501
02F1 502
02F1 503
02F1 504
02F1 505
02F1 506
02F1 507
02F1 508
02F1 509
02F1 510
02F1 511
02F1 512
02F1 513
02F1 514
02F1 515
02F1 516
02F1 517
02F1 518
02F1 519
02F1 520
02F1 521
02F1 522
02F1 523
02F1 524
02F1 525
02F1 526
02F1 527
02F1 528
02F1 529
02F1 530
02F1 531
02F1 532
02F1 533
02F1 534
02F1 535
02F1 536
02F1 537
02F1 538
02F1 539
02F1 540
02F1 541
02F1 542
02F1 543
02F1 544
02F1 545
02F1 546
02F1 547
02F1 548
02F1 549
02F1 550
02F1 551
02F1 552
02F1 553
02F1 554
02F1 555
02F1 556
02F1 557
02F1 558
02F1 559
02F1 560
02F1 561
02F1 562
02F1 563
02F1 564
02F1 565
02F1 566
02F1 567
02F1 568
02F1 569
02F1 570
02F1 571
02F1 572
02F1 573
02F1 574
02F1 575
02F1 576
02F1 577
02F1 578
02F1 579
02F1 580
02F1 581
02F1 582
02F1 583
02F1 584
02F1 585
02F1 586
02F1 587
02F1 588
02F1 589
02F1 590
02F1 591
02F1 592
02F1 593
02F1 594
02F1 595
02F1 596
02F1 597
02F1 598
02F1 599
02F1 600
02F1 601
02F1 602
02F1 603
02F1 604
02F1 605
02F1 606
02F1 607
02F1 608
02F1 609
02F1 610
02F1 611
02F1 612
02F1 613
02F1 614
02F1 615
02F1 616
02F1 617
02F1 618
02F1 619
02F1 620
02F1 621
02F1 622
02F1 623
02F1 624
02F1 625
02F1 626
02F1 627
02F1 628
02F1 629
02F1 630
02F1 631
02F1 632
02F1 633
02F1 634
02F1 635
02F1 636
02F1 637
02F1 638
02F1 639
02F1 640
02F1 641
02F1 642
02F1 643
02F1 644
02F1 645
02F1 646
02F1 647
02F1 648
02F1 649
02F1 650
02F1 651
02F1 652
02F1 653
02F1 654
02F1 655
02F1 656
02F1 657
02F1 658
02F1 659
02F1 660
02F1 661
02F1 662
02F1 663
02F1 664
02F1 665
02F1 666
02F1 667
02F1 668
02F1 669
02F1 670
02F1 671
02F1 672
02F1 673
02F1 674
02F1 675
02F1 676
02F1 677
02F1 678
02F1 679
02F1 680
02F1 681
02F1 682
02F1 683
02F1 684
02F1 685
02F1 686
02F1 687
02F1 688
02F1 689
02F1 690
02F1 691
02F1 692
02F1 693
02F1 694
02F1 695
02F1 696
02F1 697
02F1 698
02F1 699
02F1 700
02F1 701
02F1 702
02F1 703
02F1 704
02F1 705
02F1 706
02F1 707
02F1 708
02F1 709
02F1 710
02F1 711
02F1 712
02F1 713
02F1 714
02F1 715
02F1 716
02F1 717
02F1 718
02F1 719
02F1 720
02F1 721
02F1 722
02F1 723
02F1 724
02F1 725
02F1 726
02F1 727
02F1 728
02F1 729
02F1 730
02F1 731
02F1 732
02F1 733
02F1 734
02F1 735
02F1 736
02F1 737
02F1 738
02F1 739
02F1 740
02F1 741
02F1 742
02F1 743
02F1 744
02F1 745
02F1 746
02F1 747
02F1 748
02F1 749
02F1 750
02F1 751
02F1 752
02F1 753
02F1 754
02F1 755
02F1 756
02F1 757
02F1 758
02F1 759
02F1 760
02F1 761
02F1 762
02F1 763
02F1 764
02F1 765
02F1 766
02F1 767
02F1 768
02F1 769
02F1 770
02F1 771
02F1 772
02F1 773
02F1 774
02F1 775
02F1 776
02F1 777
02F1 778
02F1 779
02F1 780
02F1 781
02F1 782
02F1 783
02F1 784
02F1 785
02F1 786
02F1 787
02F1 788
02F1 789
02F1 790
02F1 791
02F1 792
02F1 793
02F1 794
02F1 795
02F1 796
02F1 797
02F1 798
02F1 799
02F1 800
02F1 801
02F1 802
02F1 803
02F1 804
02F1 805
02F1 806
02F1 807
02F1 808
02F1 809
02F1 810
02F1 811
02F1 812
02F1 813
02F1 814
02F1 815
02F1 816
02F1 817
02F1 818
02F1 819
02F1 820
02F1 821
02F1 822
02F1 823
02F1 824
02F1 825
02F1 826
02F1 827
02F1 828
02F1 829
02F1 830
02F1 831
02F1 832
02F1 833
02F1 834
02F1 835
02F1 836
02F1 837
02F1 838
02F1 839
02F1 840
02F1 841
02F1 842
02F1 843
02F1 844
02F1 845
02F1 846
02F1 847
02F1 848
02F1 849
02F1 850
02F1 851
02F1 852
02F1 853
02F1 854
02F1 855
02F1 856
02F1 857
02F1 858
02F1 859
02F1 860
02F1 861
02F1 862
02F1 863
02F1 864
02F1 865
02F1 866
02F1 867
02F1 868
02F1 869
02F1 870
02F1 871
02F1 872
02F1 873
02F1 874
02F1 875
02F1 876
02F1 877
02F1 878
02F1 879
02F1 880
02F1 881
02F1 882
02F1 883
02F1 884
02F1 885
02F1 886
02F1 887
02F1 888
02F1 889
02F1 890
02F1 891
02F1 892
02F1 893
02F1 894
02F1 895
02F1 896
02F1 897
02F1 898
02F1 899
02F1 900
02F1 901
02F1 902
02F1 903
02F1 904
02F1 905
02F1 906
02F1 907
02F1 908
02F1 909
02F1 910
02F1 911
02F1 912
02F1 913
02F1 914
02F1 915
02F1 916
02F1 917
02F1 918
02F1 919
02F1 920
02F1 921
02F1 922
02F1 923
02F1 924
02F1 925
02F1 926
02F1 927
02F1 928
02F1 929
02F1 930
02F1 931
02F1 932
02F1 933
02F1 934
02F1 935
02F1 936
02F1 937
02F1 938
02F1 939
02F1 940
02F1 941
02F1 942
02F1 943
02F1 944
02F1 945
02F1 946
02F1 947
02F1 948
02F1 949
02F1 950
02F1 951
02F1 952
02F1 953
02F1 954
02F1 955
02F1 956
02F1 957
02F1 958
02F1 959
02F1 960
02F1 961
02F1 962
02F1 963
02F1 964
02F1 965
02F1 966
02F1 967
02F1 968
02F1 969
02F1 970
02F1 971
02F1 972
02F1 973
02F1 974
02F1 975
02F1 976
02F1 977
02F1 978
02F1 979
02F1 980
02F1 981
02F1 982
02F1 983
02F1 984
02F1 985
02F1 986
02F1 987
02F1 988
02F1 989
02F1 990
02F1 991
02F1 992
02F1 993
02F1 994
02F1 995
02F1 996
02F1 997
02F1 998
02F1 999
02F1 1000
02F1 1001
02F1 1002
02F1 1003
02F1 1004
02F1 1005
02F1 1006
02F1 1007
02F1 1008
02F1 1009
02F1 1010
02F1 1011
02F1 1012
02F1 1013
02F1 1014
02F1 1015
02F1 1016
02F1 1017
02F1 1018
02F1 1019
02F1 1020
02F1 1021
02F1 1022
02F1 1023
02F1 1024
02F1 1025
02F1 1026
02F1 1027
02F1 1028
02F1 1029
02F1 1030
02F1 1031
02F1 1032
02F1 1033
02F1 1034
02F1 1035
02F1 1036
02F1 1037
02F1 1038
02F1 1039
02F1 1040
02F1 1041
02F1 1042
02F1 1043
02F1 1044
02F1 1045
02F1 1046
02F1 1047
02F1 1048
02F1 1049
02F1 1050
02F1 1051
02F1 1052
02F1 1053
02F1 1054
02F1 1055
02F1 1056
02F1 1057
02F1 1058
02F1 1059
02F1 1060
02F1 1061
02F1 1062
02F1 1063
02F1 1064
02F1 1065
02F1 1066
02F1 1067
02F1 1068
02F1 1069
02F1 1070
02F1 1071
02F1 1072
02F1 1073
02F1 1074
02F1 1075
02F1 1076
02F1 1077
02F1 1078
02F1 1079
02F1 1080
02F1 1081
02F1 1082
02F1 1083
02F1 1084
02F1 1085
02F1 1086
02F1 1087
02F1 1088
02F1 1089
02F1 1090
02F1 1091
02F1 1092
02F1 1093
02F1 1094
02F1 1095
02F1 1096
02F1 1097
02F1 1098
02F1 1099
02F1 1100
02F1 1101
02F1 1102
02F1 1103
02F1 1104
02F1 1105
02F1 1106
02F1 1107
02F1 1108
02F1 1109
02F1 1110
02F1 1111
02F1 1112
02F1 1113
02F1 1114
02F1 1115
02F1 1116
02F1 1117
02F1 1118
02F1 1119
02F1 1120
02F1 1121
02F1 1122
02F1 1123
02F1 1124
02F1 1125
02F1 1126
02F1 1127
02F1 1128
02F1 1129
02F1 1130
02F1 1131
02F1 1132
02F1 1133
02F1 1134
02F1 1135
02F1 1136
02F1 1137
02F1 1138
02F1 1139
02F1 1140
02F1 1141
02F1 1142
02F1 1143
02F1 1144
02F1 1145
02F1 1146
02F1 1147
02F1 1148
02F1 1149
02F1 1150
02F1 1151
02F1 1152
02F1 1153
02F1 1154
02F1 1155
02F1 1156
02F1 1157
02F1 1158
02F1 1159
02F1 1160
02F1 1161
02F1 1162
02F1 1163
02F1 1164
02F1 1165
02F1 1166
02F1 1167
02F1 1168
02F1 1169
02F1 1170
02F1 1171
02F1 1172
02F1 1173
02F1 1174
02F1 1175
02F1 1176
02F1 1177
02F1 1178
02F1 1179
02F1 1180
02F1 1181
02F1 1182
02F1 1183
02F1 1184
02F1 1185
02F1 1186
02F1 1187
02F1 1188
02F1 1189
02F1 1190
02F1 1191
02F1 1192
02F1 1193
02F1 1194
02F1 1195
02F1 1196
02F1 1197
02F1 1198
02F1 1199
02F1 1200
02F1 1201
02F1 1202
02F1 1203
02F1 1204
02F1 1205
02F1 1206
02F1 1207
02F1 1208
02F1 1209
02F1 1210
02F1 1211
02F1 1212
02F1 1213
02F1 1214
02F1 1215
02F1 1216
02F1 1217
02F1 1218
02F1 1219
02F1 1220
02F1 1221
02F1 1222
02F1 1223
02F1 1224
02F1 1225
02F1 1226
02F1 1227
02F1 1228
02F1 1229
02F1 1230
02F1 1231
02F1 1232
02F1 1233
02F1 1234
02F1 1235
02F1 1236
02F1 1237
02F1 1238
02F1 1239
02F1 1240
02F1 1241
02F1 1242
02F1 1243
02F1 1244
02F1 1245
02F1 1246
02F1 1247
02F1 1248
02F1 1249
02F1 1250
02F1 1251
02F1 1252
02F1 1253
02F1 1254
02F1 1255
02F1 1256
02F1 1257
02F1 1258
02F1 1259
02F1 1260
02F1 1261
02F1 1262
02F1 1263
02F1 1264
02F1 1265
02F1 1266
02F1 1267
02F1 1268
02F1 1269
02F1 1270
02F1 1271
02F1 1272
02F1 1273
02F1 1274
02F1 1275
02F1 1276
02F1 1277
02F1 1278
02F1 1279
02F1 1280
02F1 1281
02F1 1282
02F1 1283
02F1 1284
02F1 1285
02F1 1286
02F1 1287
02F1 1288
02F1 1289
02F1 1290
02F1 1291
02F1 1292
02F1 1293
02F1 1294
02F1 1295
02F1 1296
02F1 1297
02F1 1298
02F1 1299
02F1 1300
02F1 1301
02F1 1302
02F1 1303
02F1 1304
02F1 1305
02F1 1306
02F1 1307
02F1 1308
02F1 1309
02F1 1310
02F1 1311
02F1 1312
02F1 1313
02F1 1314
02F1 1315
02F1 1316
02F1 1317
02F1 1318
02F1 1319
02F1 1320
02F1 1321
02F1 1322
02F1 1323
02F1 1324
02F1 1325
02F1 1326
02F1 1327
02F1 1328
02F1 1329
02F1 1330
02F1 1331
02F1 1332
02F1 1333
02F1 1334
02F1 1335
02F1 1336
02F1 1337
02F1 1338
02F1 1339
02F1 1340
02F1 1341
02F1 1342
02F1 1343
02F1 1344
02F1 1345
02F1 1346
02F1 1347
02F1 1348
02F1 1349
02F1 1350
02F1 1351
02F1 1352
02F1 1353
02F1 1354
02F1 1355
02F1 1356
02F1 1357
02F1 1358
02F1 1359
02F1 1360
02F1 1361
02F1 1362
02F1 1363
02F1 1364
02F1 1365
02F1 1366
02F1 1367
02F1 1368
02F1 1369
02F1 1370
02F1 1371
02F1 1372
02F1 1373
02F1 1374
02F1 1375
02F1 1376
02F1 1377
02F1 1378
02F1 1379
02F1 1380
02F1 1381
02F1 1382
02F1 1383
02F1 1384
02F1 1385
02F1 1386
02F1 1387
02F1 1388
02F1 1389
02F1 1390
02F1 1391
02F1 1392
02F1 1393
02F1 1394
02F1 1395
02F1 1396
02F1 1397
02F1 1398
02F1 1399
02F1 1400
02F1 1401
02F1 1402
02F1 1403
02F1 1404
02F1 1405
02F1 1406
02F1 1407
02F1 1408
02F1 1409
02F1 1410
02F1 1411
02F1 1412
02F1 1413
02F1 1414
02F1 1415
02F1 1416
02F1 1417
02F1 1418
02F1 1419
02F1 1420
02F1 1421
02F1 1422
02F1 1423
02F1 1424
02F1 1425
02F1 1426
02F1 1427
02F1 1428
02F1 1429
02F1 1430
02F1 1431
02F1 1432
02F1 1433
02F1 1434
02F1 1435
02F1 1436
02F1 1437
02F1 1438
02F1 1439
02F1 1440
02F1 1441
02F1 1442
02F1 1443
02F1 1444
02F1 1445
02F1 1446
02F1 1447
02F1 1448
02F1 1449
02F1 1450
02F1 1451
02F1 1452
02F1 1453
02F1 1454
02F1 1455
02F1 1456
02F1 1457
02F1 1458
02F1 1459
02F1 1460
02F1 1461
02F1 1462
02F1 1463
02F1 1464
02F1 1465
02F1 1466
02F1 1467
02F1 1468
02F1 1469
02F1 1470
02F1 1471
02F1 1472
02F1 1473
02F1 1474
02F1 1475
02F1 1476
02F1 1477
02F1 1478
02F1 1479
02F1 1480
02F1 1481
02F1 1482
02F1 1483
02F1 1484
02F1 1485
02F1 1486
02F1 1487
02F1 1488
02F1 1489
02F1 1490
02F1 1491
02F1 1492
02F1 1493
02F1 1494
02F1 1495
02F1 1496
02F1 1497
02F1 1498
02F1 1499
02F1 1500
02F1 1501
02F1 1502
02F1 1503
02F1 1504
02F1 1505
02F1 1506
02F1 1507
02F1 1508
02F1 1509
02F1 1510
02F1 1511
02F1 1512
02F1 1513
02F1 1514
02F1 1515
02F1 1516
02F1 1517
02F1 1518
02F1 1519
02F1 1520
02F1 1521
02F1 1522
02F1 1523
02F1 1524
02F1 1525
02F1 1526
02F1 1527
02F1 1528
02F1 1529
02F1 1530
02F1 1531
02F1 1532
02F1 1533
02F1 1534
02F1 1535
02F1 1536
02F1 1537
02F1 1538
02F1 1539
02F1 1540
02F1 1541
02F1 1542
02F1 1543
02F1 1544
02F1 1545
02F1 1546
02F1 1547
02F1 1548
02F1 1549
02F1 1550
02F1 1551
02F1 1552
02F1 1553
02F1 1554
02F1 1555
02F1 1556
02F1 1557
02F1 1558
02F1 1559
02F1 1560
02F1 1561
02F1 1562
02F1 1563
02F1 1564
02F1 1565
02F1 1566
02F1 1567
02F1 1568
02F1 1569
02F1 1570
02F1 1571
02F1 1572
02F1 1573
02F1 1574
02F1 1575
02F1 1576
02F1 1577
02F1 1578
02F1 1579
02F1 1580
02F1 1581
02F1 1582
02F1 1583
02F1 1584
02F1 1585
02F1 1586
02F1 1587
02F1 1588
02F1 1589
02F1 1590
02F1 1591
02F1 1592
02F1 1593
02F1 1594
02F1 1595
02F1 1596
02F1 1597
02F1 1598
02F1 1599
02F1 1600
02F1 1601
02F1 1602
02F1 1603
02F1 1604
02F1 1605
02F1 1606
02F1 1607
02F1 1608
02F1 1609
02F1 1610
02F1 1611
02F1 1612
02F1 1613
02F1 1614
02F1 1615
02F1 1616
02F1 1617
02F1 1618
02F1 1619
02F1 1620
02F1 1621
02F1 1622
02F1 1623
02F1 
```


Character formatting routines

```

      72 15 02F1 472 BLEQ OUT_ROUND : Done if no more sig. digits
      AC AB D7 02F3 473 DECL LONG_COUNT(R8) : Decrement longword count
50 AC AB D0 02F6 474 MOVL LONG_COUNT(R8), R0
      03 18 02FA 475 BGEQ 1$
      00F8 31 02FC 476 BRW OUT_FRACT : Do fraction part if time
55 09 C0 02FF 477 1$: ADDL2 #9, R5 : R5 will store least signif digit
      53 55 D0 0302 478 : (lsd) in the high order byte.
      F7 A5 FCF7 CF 7D 0305 479 : save the old address
      51 B4 AB40 D0 0308 480 MOVQ ASCII_ZEROES, -9(R5) : Initialize the string to contain 30's
      OA 51 D1 030B 481 : the 9th byte will be filled below
      44 19 0310 482 MOVL DIGITS(R8)[R0], R1
      52 D4 0313 483 CMPL R1, #X000000A : if R1 < 10 you may skip the EDIV
54 51 51 00000064 8F 7B 0315 484 BLSS 70$ :
      31 13 0317 485 CLRL R2 : R1/R2 must be a quadword for the EDIV
      75 FCE1 CF44 B0 0322 486 EDIV #100, R1, R1, R4 : extract two lsd
54 51 51 00000064 8F 7B 0328 487 BEQL 60$ :
      20 13 0331 488 MOVW TABLE[R4], -(R5) : load correct char rep of the 2 digits
      75 FC00 CF44 B0 0333 489 EDIV #100, R1, R1, R4 : extract two lsd
54 51 51 00000064 8F 7B 0339 490 BEQL 60$ :
      OF 13 0342 491 MOVW TABLE[R4], -(R5) : load correct char rep of the 2 digits
54 51 51 00000064 8F 7B 0339 492 EDIV #100, R1, R1, R4 : extract two lsd
      75 FCBF CF44 B0 0344 493 BEQL 60$ :
      75 FC00 CF44 B0 034A 494 MOVW TABLE[R4], -(R5) : load correct char rep of the 2 digits
      51 30 B0 0353 495 EDIV #100, R1, R1, R4 : extract two lsd
      55 53 B0 0359 496 60$: MOVW TABLE[R4], -(R5) : load correct char rep of the 2 digits
      56 09 C2 035D 497 70$: ADDB3 #A/0/, R1, -(R5) : character rep needed for last number
      8C 11 0360 498 MOVL R3, R5 : Move string pointer up by 9
      0157 31 0363 499 SUBL2 #9, R6 : Adjust # of sig. digits
      501 BRB OUT_LOOP
      502 OUT_ROUND:
      503 : BRB ROUND
      504 : BRW ROUND
      505
      506 :+
      507 : This code is executed if the value is less than 1.
      508 :-
      509 NO_INT:
      510 BSBW INIT_FRACT : Initialize the pointers
      511 : and get first 9 digits.
      512 E4 A7 D4 036B 513 10$: CLRL DEC_EXP(R7) : Calculate exponent
      514 09 C2 036E 514 : #9, DEC_EXP(R7) : Its 9 smaller now
      515 51 D5 0372 515 : TSTL R1 : Are digits zero?
      516 05 12 0374 516 BNEQ 20$
      517 FE9C 30 0376 517 BSBW GET_FRACT : Get next 9 digits
      518 F3 11 0379 518 BRB 10$ : And try again
      519 55 09 C0 037B 519 20$: ADDL2 #9, R5 : R5 will store least signif digit
      520 037E 520 : (lsd) in the high order byte.
      521 F7 A5 FCF7 CF 7D 0381 521 MOVQ ASCII_ZEROES, -9(R5) : save the old address
      522 OA 51 D1 0387 522 : Initialize the string to contain 30's
      523 44 19 038A 523 : the 9th byte will be filled below
      524 52 D4 038C 524 CMPL R1, #X000000A : if R1 < 10 you may skip the EDIV
      525 51 13 0397 525 CLRL R2 : R1/R2 must be a quadword for the EDIV
54 51 51 00000064 8F 7B 038E 526 EDIV #100, R1, R1, R4 : extract two lsd
      31 13 0397 527 BEQL 60$ :
      75 FC6A CF44 B0 0399 528 MOVW TABLE[R4], -(R5) : load correct char rep of the 2 digits
```

Character formatting routines

```
54 51 51 00000064 8F 7B 039F 529 EDIV #100, R1, R1, R4 ; extract two lsd
      20 13 03A8 530 BEQL 60$
      75 FC59 CF44 B0 03AA 531 MOVW TABLE[R4], -(R5) ; load correct char rep of the 2 digits
54 51 51 00000064 8F 7B 03B0 532 EDIV #100, R1, R1, R4 ; extract two lsd
      OF 13 03B9 533 BEQL 60$
      75 FC48 CF44 B0 03BB 534 MOVW TABLE[R4], -(R5) ; load correct char rep of the 2 digits
54 51 51 00000064 8F 7B 03C1 535 EDIV #100, R1, R1, R4 ; extract two lsd
      75 FC39 CF44 B0 03CA 536 60$: MOVW TABLE[R4], -(R5) ; load correct char rep of the 2 digits
      75 51 30 81 03D0 537 70$: ADDB3 #^A/0/, R1, -(R5) ; character rep needed for last number
      03D4 538
      03D4 539
      03D4 540
      03D4 541
      03D4 542
      03D4 543
      03D4 544
      03D4 545
      03D4 546
      03D4 547
      50 53 55 C3 03D4 548
      51 D5 03D8 549
      0A 12 03DA 550
      50 D7 03DC 551
      30 01 A5 91 03DE 552
      02 12 03E2 553
      50 D7 03E4 554
      03E6 555
      03E6 556
      E0 A7 0A 50 C3 03E6 557 98$: SUBL3 R5, R3, R0
      E4 A7 50 C0 03EB 558 ADDL2 R0, DEC_EXP(R7)
      55 53 D0 03EF 559 MOVL R3, R5
      56 50 C2 03F2 560 SUBL2 R0, R6
      62 11 03F5 561 BRB FRAC_LOOP
      03F7 562
      03F7 563
      03F7 564
      03F7 565
      03F7 566
      FE0D 30 03F7 567
      55 09 C0 03FA 568
      03FD 569
      53 55 D0 03FD 570
      F7 A5 FBFC CF 7D 0400 571
      0406 572
      0A 51 D1 0406 573
      44 19 0409 574
      52 D4 040B 575
54 51 51 00000064 8F 7B 040D 576 EDIV #100, R1, R1, R4 ; extract two lsd
      31 13 0416 577 BEQL 60$
      75 FBEB CF44 B0 0418 578 MOVW TABLE[R4], -(R5) ; load correct char rep of the 2 digits
54 51 51 00000064 8F 7B 041E 579 EDIV #100, R1, R1, R4 ; extract two lsd
      20 13 0427 580 BEQL 60$
      75 FBDA CF44 B0 0429 581 MOVW TABLE[R4], -(R5) ; load correct char rep of the 2 digits
54 51 51 00000064 8F 7B 042F 582 EDIV #100, R1, R1, R4 ; extract two lsd
      OF 13 0438 583 BEQL 60$
      75 FBC9 CF44 B0 043A 584 MOVW TABLE[R4], -(R5) ; load correct char rep of the 2 digits
54 51 51 00000064 8F 7B 0440 585 EDIV #100, R1, R1, R4 ; extract two lsd
```

Numbers are stored as characters as follows: low order byte is the most significant digit (character), while the high order byte is the least significant digit (character). The storage took place from the high order digit to the low order digit. Since we used an EDIV by 100, 0, 1, or 2 zeroes may be located at (R5). R0 is to contain the number of nonzero digits (not char 30) between (R3) and (R5). If R1<>0 then R0=9. If R1=0, there is at least one zero at (R5) and possibly another at (R5)-1. For example, 12 --> 323130 while 102 --> 3230313030.

At least on leading 30.
there still could be 1 more 0
102 --> 3230313030 by the above
we've already seen rightmost 30
if there is another, subt 1.
There can be no more consec 0's

Calculate exponent
Move string pointer up by 9
Adjust # of sig. digits
Get 9 more

++
This code starts the fraction portion if the integer portion exists.
--
OUT_FRACT:

Initialize and get 9 digits
R5 will store least signif digit
(lsd) in the high order byte.
save the old address
Initialize the string to contain 30's
the 9th byte will be filled below
if R1 < 10 you may skip the EDIV

R1/R2 must be a quadword for the EDIV
extract two lsd

load correct char rep of the 2 digits
extract two lsd

load correct char rep of the 2 digits
extract two lsd

load correct char rep of the 2 digits
extract two lsd

Character formatting routines

```
75 FBBA CF44 B0 0449 586 60$: MOVW TABLE[R4], -(R5) ; load correct char rep of the 2 digits
75 51 30 81 044F 587 70$: ADDB3 #^A/0/, R1, -(R5) ; character rep needed for last number
55 53 D0 0453 588 ; MOVL R3, R5 ; Move string pointer up by 9
56 09 C2 0456 589 ; SUBL2 #9, R6 ; Adjust # of sig. digits
590 FRACT_LOOP:
64 15 0459 591 ; BLEQ ROUND ; If not, finish
55 FDB7 30 045B 592 ; BSBW GET_FRACT ; Get 9 more digits
09 C0 045E 593 ; ADDL2 #9, R5 ; R5 will store least signif digit
53 55 D0 0461 594 ; ; (lsd) in the high order byte.
F7 A5 FB98 CF 7D 0464 595 ; MOVL R5, R3 ; save the old address
0A 51 D1 046A 596 ; MOVQ ASCII_ZEROES, -9(R5) ; Initialize the string to contain 30's
44 19 046D 597 ; ; the 9th byte will be filled below
52 D4 046F 600 ; CMPL R1, #^X000000A ; if R1 < 10 you may skip the EDIV
54 51 51 00000064 8F 7B 0471 601 ; BLSS 70$ ; R1/R2 must be a quadword for the EDIV
31 13 047A 602 ; CLRL R2 ; extract two lsd
75 FB87 CF44 B0 047C 603 ; MOVW TABLE[R4], -(R5) ; load correct char rep of the 2 digits
54 51 51 00000064 8F 7B 0482 604 ; EDIV #100, R1, R1, R4 ; extract two lsd
20 13 048B 605 ; BEQL 60$
75 FB76 CF44 B0 048D 606 ; MOVW TABLE[R4], -(R5) ; load correct char rep of the 2 digits
54 51 51 00000064 8F 7B 0493 607 ; EDIV #100, R1, R1, R4 ; extract two lsd
0F 13 049C 608 ; BEQL 60$
75 FB65 CF44 B0 049E 609 ; MOVW TABLE[R4], -(R5) ; load correct char rep of the 2 digits
54 51 51 00000064 8F 7B 04A4 610 ; EDIV #100, R1, R1, R4 ; extract two lsd
75 FB56 CF44 B0 04AD 611 60$: MOVW TABLE[R4], -(R5) ; load correct char rep of the 2 digits
75 51 30 81 04B3 612 70$: ADDB3 #^A/0/, R1, -(R5) ; character rep needed for last number
55 53 D0 04B7 613 ; MOVL R3, R5 ; Move string pointer up by 9
56 09 C2 04BA 614 ; SUBL2 #9, R6 ; Adjust # of sig. digits
9A 11 04BD 615 ; BRB FRACT_LOOP ; Loop back for more
04BF 616
04BF 617 ;+
04BF 618 ; This routine rounds the value to the given number of significant
04BF 619 ; digits, unless flag V_TRUNCATE is on. If so, the value is truncated
04BF 620 ; at the next digit.
04BF 621 ; -
04BF 622 ROUND:
04BF 623 DECL R6
04BF 624 ADDL2 R6, R5 ; Find least significant + 1
41 F4 A7 18 E0 04C4 625 ; BBS #V_TRUNCATE, FLAGS(R7), FINIS ; Truncate if desired
17 F4 A7 19 E1 04C9 626 ; BBC #V_ROUND_RIGHT, FLAGS(R7), 5$ ; Round to right of dec pt?
50 DC A7 E4 A7 C1 04CE 627 ; ADDL3 DEC_EXP(R7), R1_RND(R7), R0 ; Yes, find it
34 19 04D4 628 ; BLSS FINIS ; Done if rounds to zero
F0 A7 50 D1 04D6 629 ; CMPL R0, SIG_DIGITS(R7) ; Round to right of # sig digits?
09 18 04DA 630 ; BGEQ 5$ ; Yes, round to significant digits
50 50 E0 A7 C0 04DC 631 ; ADDL2 OFFSET(R7), R0 ; Finish calculation
55 EC A7 50 C1 04E0 632 ; ADDL3 R0, STRING_ADDR(R7), R5 ; Get rounding character address
35 65 91 04E5 633 5$: CMPB (R5), #^A/5/ ; Round?
20 19 04E8 634 ; BLSS FINIS ; No, just finish
50 55 D0 04EA 635 ; MOVL R5, R0 ; Save position
39 70 91 04ED 636 10$: CMPB -(R0), #^A/9/ ; If this is a 9...
05 19 04F0 637 ; BLSS 20$
60 30 90 04F2 638 ; MOVB #^A/0/, (R0) ; Then it becomes a zero
F6 11 04F5 639 ; BRB 10$ ; And we continue
50 60 96 04F7 640 20$: INCB (R0) ; Else this is last carry
E0 A7 50 D1 04F9 641 ; SUBL2 STRING_ADDR(R7), R0 ; Do we need to change offset
E0 A7 50 D1 04FD 642 ; CMPL R0, OFFSET(R7) ; and exponent?
```


Character formatting routines

```

      E0 A7 07 18 0501 643      BGEQ  FINIS      ; No
      E4 A7 50 D0 0503 644      MOVL  R0, OFFSET(R7) ; Yes, set new offset
      E4 A7 D6 D6 0507 645      INCL  DEC_EXP(R7)  ; Set new exponent
      050A 646
      050A 647 ;+
      050A 648 ; ALL done.
      050A 649 ;+
      050A 650 FINIS:
SE  FFFFFFFFAB 8F C2 050A 651      SUBL2 #LOCAL_FRAME, SP ; Restore stack pointer
      51 57 D0 0511 652      MOVL  R7, R1- ; Restore common frame pointer
      05 05 0514 653      RSB ; Return to caller
      0515 654
      0515 655      .END

```

OTSS\$CVTDT
Symbol table

B 13

16-SEP-1984 00:23:22 VAX/VMS Macro V04-00
6-SEP-1984 11:12:52 [LIBRTL.SRC]OTSCVTDT.MAR;1

Page 16
(7)

ASCII_ZEROES		00000000	R	01
BIN_EXP	=	FFFFFFF80		
BIN_PT	=	FFFFFFFE4		
COMMON_FD		000000EC	R	01
DEC_EXP	=	FFFFFFFE4		
DIGITS	=	FFFFFFB4		
EXTRACT		000000F4	R	01
FINIS		0000050A	R	01
FLAGS	=	FFFFFFF4		
FORMAT		0000025C	R	01
FOUR_LONG		0000017C	R	01
FRACT_LIM	=	FFFFFFFC8		
FRACT_LOOP		00000459	R	01
FRACT_ONLY		00000175	R	01
GET_FRACT		00000215	R	01
INIT_FRACT		00000207	R	01
INT_DIV		0000019B	R	01
INT_HI	=	FFFFFFF0		
INT_LOOP		00000187	R	01
INT_NEXT		000001DE	R	01
LOCAL_FRAME	=	FFFFFFFA8		
LONG_COUNT	=	FFFFFFFAC		
NO_INT		00000368	R	01
OFFSET	=	FFFFFFFE0		
ONE_LONG		00000160	R	01
OTSS\$CVT_D_T_R8		000000DA	RG	01
OTSS\$CVT_F_T_R8		000000D0	RG	01
OUT_FRACT		000003F7	R	01
OUT_LOOP		000002F1	R	01
OUT_ROUND		00000365	R	01
PACKED	=	FFFFFFF8		
ROUND		000004BF	R	01
RT_RND	=	FFFFFFDC		
SIGN	=	FFFFFFE8		
SIG_DIGITS	=	FFFFFFF0		
STRING_ADDR	=	FFFFFFEC		
TABLE		00000008	R	01
TEMP	=	FFFFFFFA8		
VAL_NEG		000000EA	R	01
VAL_POS		000000F0	R	01
V_ROUND_RIGHT	=	00000019		
V_TRUNCATE	=	00000018		
ZERO		00000163	R	01

+-----+
! Psect synopsis !
+-----+

PSECT name	Allocation	PSECT No.	Attributes										
ABS	00000000 (0.)	00 (0.)	NOPIC	USR	CON	ABS	LCL	NOSHR	NOEXE	NORD	NOWRT	NOVEC	BYTE
_OTSS\$CODE	00000515 (1301.)	01 (1.)	PIC	USR	CON	REL	LCL	SHR	EXE	RD	NOWRT	NOVEC	LONG

! Performance indicators !

Phase	Page faults	CPU Time	Elapsed Time
Initialization	33	00:00:00.02	00:00:01.62
Command processing	135	00:00:00.30	00:00:02.78
Pass 1	94	00:00:01.28	00:00:07.73
Symbol table sort	0	00:00:00.03	00:00:00.03
Pass 2	130	00:00:00.86	00:00:04.28
Symbol table output	5	00:00:00.03	00:00:00.34
Psect synopsis output	2	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	401	00:00:02.54	00:00:16.80

The working set limit was 1200 pages.
13641 bytes (27 pages) of virtual memory were used to buffer the intermediate code.
There were 10 pages of symbol table space allocated to hold 44 non-local and 31 local symbols.
655 source lines were read in Pass 1, producing 10 object records in Pass 2.
0 pages of virtual memory were used to define 0 macros.

! Macro library statistics !

Macro library name	Macros defined
_\$255\$DUA28:[SYSLIB]STARLET.MLB;2	0

0 GETS were required to define 0 macros.

There were no errors, warnings or information messages.

MACRO/ENABLE=SUPPRESSION/DISABLE=(GLOBAL,TRACEBACK)/LIS=LIS\$:OTSCVTD.T/OBJ=OBJ\$:OTSCVTD.T MSRC\$:OTSCVTD.T/UPDATE=(ENH\$:OTSCVTD.T)

0211 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

